Database Programming (Part 2)

Robert M. Dondero, Ph.D.
Princeton University

Objectives

- We will cover:
  - **Databases** (**DBs**) and **database management systems** (**DBMSs**)…
  - With a focus on **relational** DBs and DBMSs…
  - With a focus on the **SQLite** DBMS…
  - With a focus on **programming** with SQLite

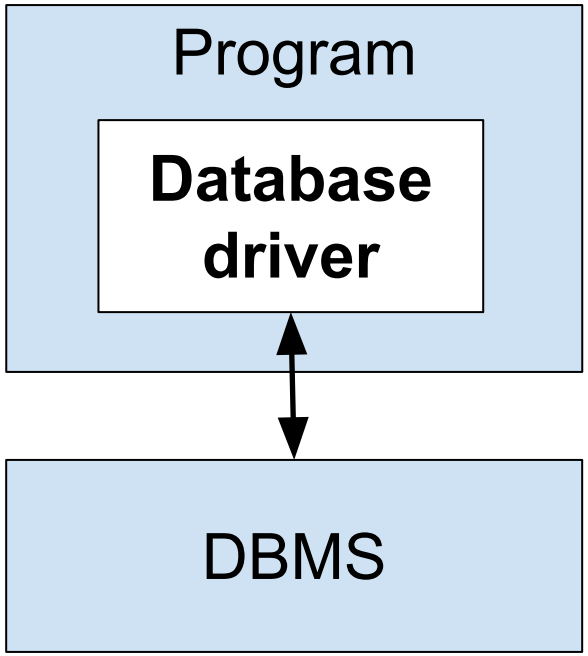
Objectives

- **Question**: How does one use SQLite?
- **Answer**: In this course:
  - Via the SQLite command-line client
  - Via programs that you compose…

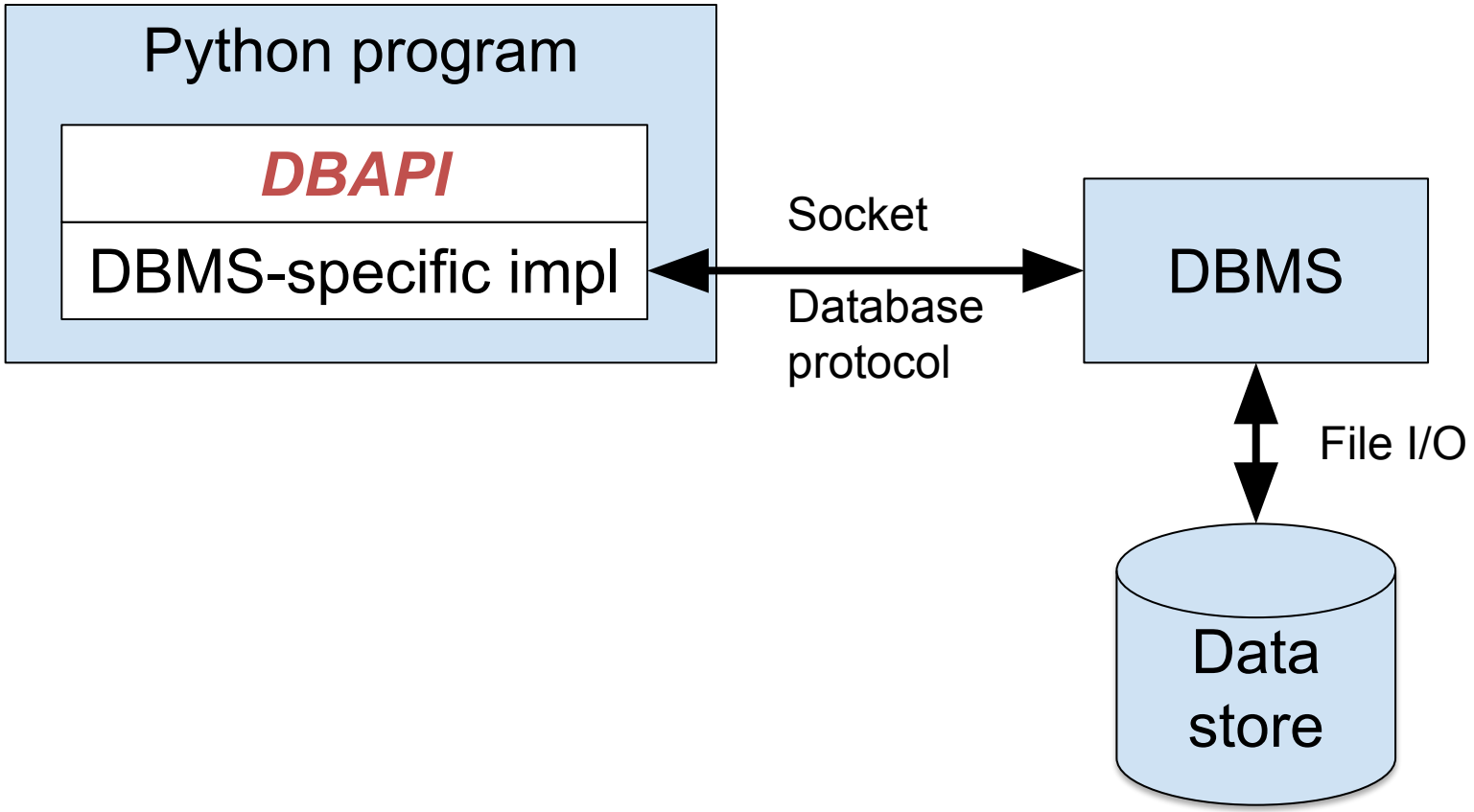
Agenda

- **Relational DB app architecture**
- Relational DB pgmming
- Relational DB pgmming: prepared stmts

Relational DB App Arch

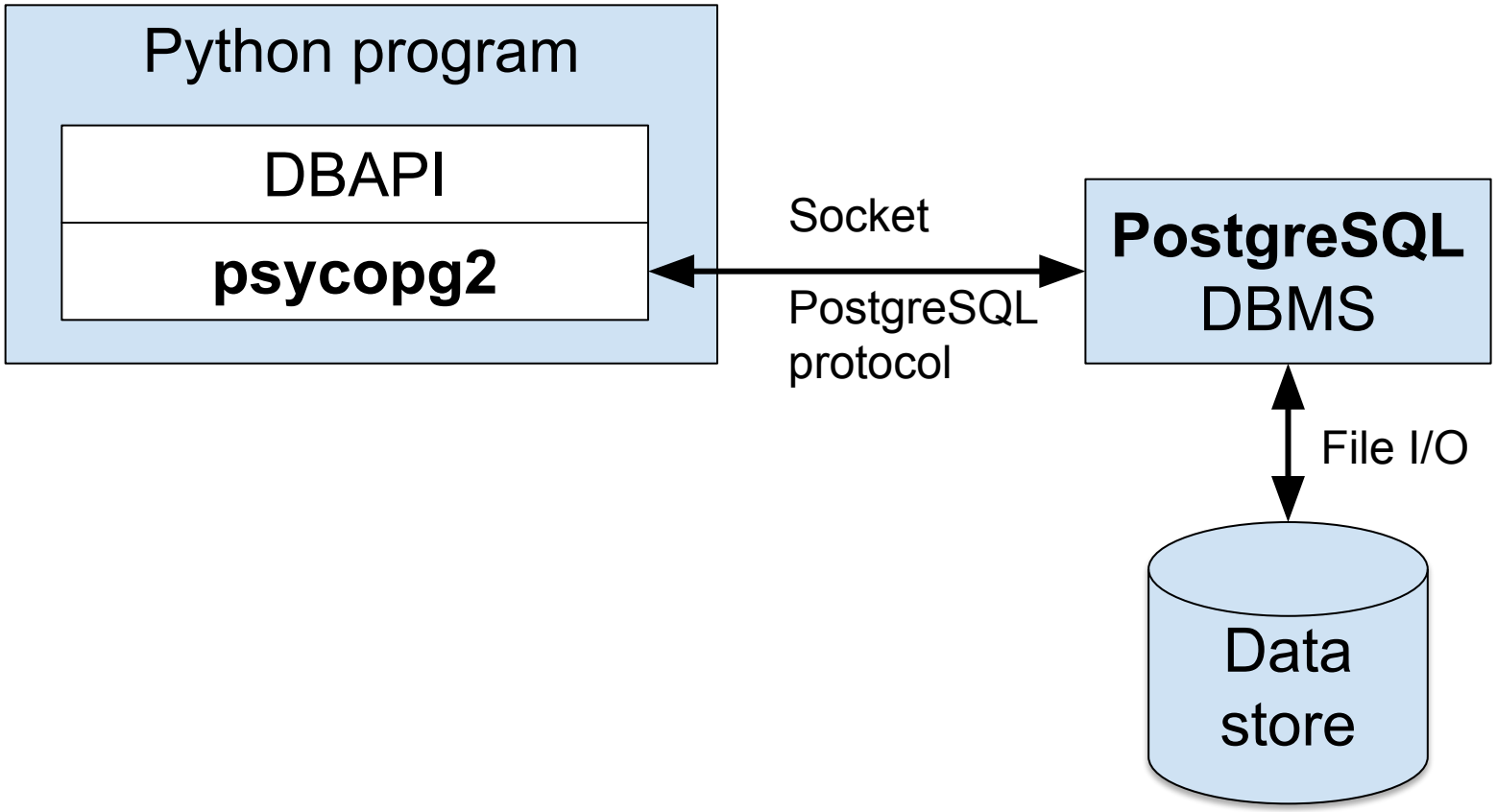


Relational DB App Arch



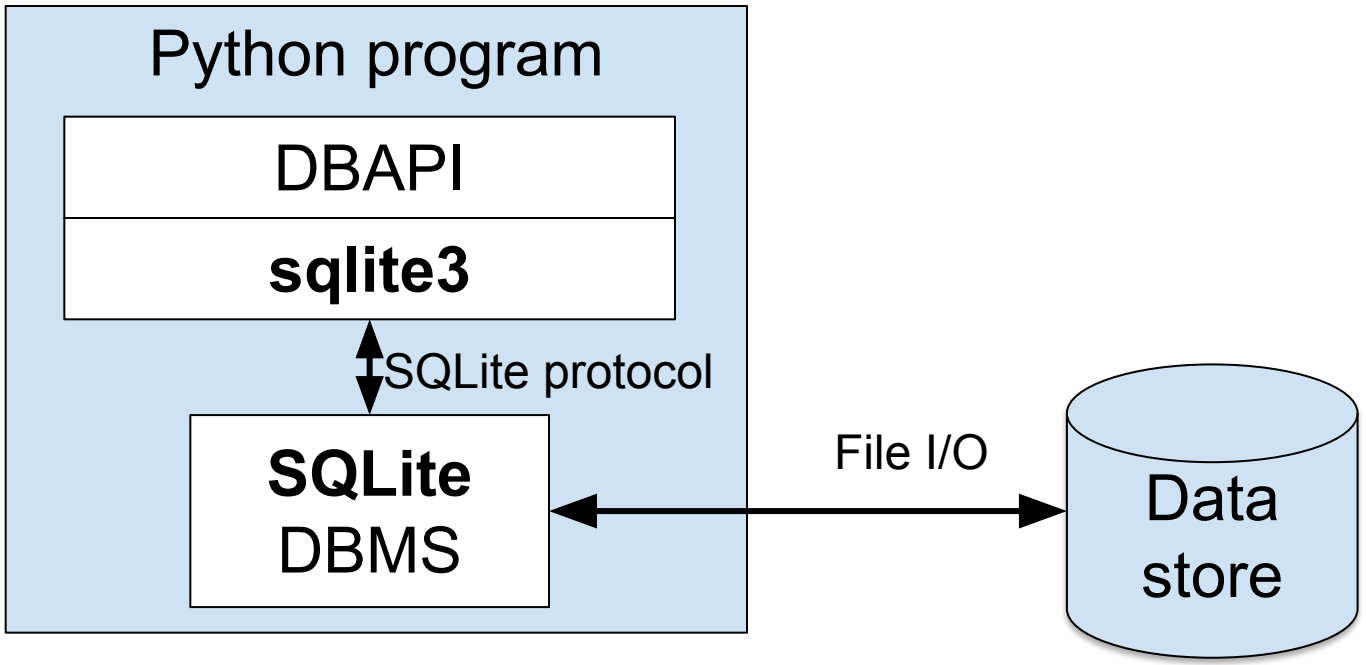
Relational DB App Arch

When using PostgreSQL



Relational DB App Arch

When using SQLite



Agenda

- Relational DB app architecture
- **Relational DB pgmming**
- Relational DB pgmming: prepared stmts

Relational DB Programming

- Code structure

Baseline:

```
connection = connect(…)
…
…
connection.close()
```

Better:

```
connection = connect(…)
try:
    …
    …
finally:
    connection.close()
```

Better still:

```
with connect(…) as connection:
    …
    …
```

Relational DB Programming

- Code structure

Baseline:

```
cursor = connection.cursor()
…
…
cursor.close()
```

Better:

```
cursor = connection.cursor(…)
try:
    …
    …
finally:
    cursor.close()
```

Better still:

```
with connection.cursor() as cursor
    …
    …
```

Relational DB Programming

- Code structure

  Better still:

```
with connection.cursor() as cursor:
    …
    …
```

```
import contextlib
…
with contextlib.closing(connection.cursor()) as cursor:
    …
    …
```

For more info: https://realpython.com/python-with-statement/

Relational DB Programming

- See **create.py**

```
$ python create.py
$
```

Relational DB Programming

- See **display.py**

```
$ python display.py
-----------------------------------
books
-----------------------------------
('123', 'The Practice of Programm
('234', 'The C Programming Langua
('345', 'Algorithms in C', 650)
-----------------------------------
authors
-----------------------------------
('123', 'Kernighan')
('123', 'Pike')
('234', 'Kernighan')
('234', 'Ritchie')
('345', 'Sedgewick')
```

```
-----------------------------------------
customers
-----------------------------------------
('111', 'Princeton', '114 Nassau St',
'08540')
('222', 'Harvard', '1256 Mass Ave',
'02138')
('333', 'MIT', '292 Main St', '02142')

-----------------------------------------
zipcodes
-----------------------------------------
('08540', 'Princeton', 'NJ')
('02138', 'Cambridge', 'MA')
('02142', 'Cambridge', 'MA')
-----------------------------------------
orders
-----------------------------------------
('123', '222', 20)
('345', '222', 100)
('123', '111', 30)
$
```

Relational DB Programming

- See **display.py** (cont.)
  - To use a cursor to select data:

```
cursor.execute("SELECT * FROM books")
table = cursor.fetchall()
for row in table:
    print(row)
```

or

```
cursor.execute("SELECT * FROM books")
row = cursor.fetchone()
while row is not None:
    print(row)
    row = cursor.fetchone()
```

Relational DB Programming

- See **authorsearch.py**

```
$ python authorsearch.py Kernighan
ISBN: 123
Title: The Practice of Programming
Quantity: 500

ISBN: 234
Title: The C Programming Language
Quantity: 800

$
```

Relational DB Programming

- See **order.py**

```
$ python display.py
…
-----------------------------------------
orders
-----------------------------------------
('123', '222', 20)
('345', '222', 100)
('123', '111', 30)
$ python order.py 123 222
$ python display.py
…
-----------------------------------------
orders
-----------------------------------------
('123', '222', 21)
('345', '222', 100)
('123', '111', 30)
$
```

Agenda

- Relational DB app architecture
- Relational DB pgmming
- **Relational DB pgmming: prepared stmts**

DB Pgmming: Prepared Stmts

- Malicious user provides this `someauthor` to authorsearch.py

```
junk' OR 'x'='x
```

DB Pgmming: Prepared Stmts

```
SELECT books.isbn, title, quantity
   FROM books, authors
   WHERE books.isbn = authors.isbn
   AND author = 'someauthor'
```

```
junk' OR 'x'='x
```

```
SELECT books.isbn, title, quantity
   FROM books, authors
   WHERE books.isbn = authors.isbn
   AND author = 'junk' OR 'x'='x'
```

AND has higher precedence than OR

```
SELECT books.isbn, title, quantity
   FROM books, authors
   WHERE (books.isbn = authors.isbn
   AND author = 'junk') OR ('x'='x')
```

DB Pgmming: Prepared Stmts

SQL injection attack

```
$ python authorsearch.py "junk' OR 'x'='x"
ISBN: 123
Title: The Practice of Pro
Quantity: 500

ISBN: 123
Title: The Practice of Pro
Quantity: 500

ISBN: 123
Title: The Practice of Pro
Quantity: 500

ISBN: 123
Title: The Practice of Pro
Quantity: 500

ISBN: 123
Title: The Practice of Pro
Quantity: 500
```

```
ISBN: 234
Title: The C Programming Language
Quantity: 800

ISBN: 234
Title: The C Programming
Quantity: 800

ISBN: 234
Title: The C Programming
Quantity: 800

ISBN: 234
Title: The C Programming
Quantity: 800

ISBN: 234
Title: The C Programming
Quantity: 800
```

```
ISBN: 345
Title: Algorithms in C
Quantity: 650

ISBN: 345
Title: Algorithms in C
Quantity: 650

ISBN: 345
Title: Algorithms in C
Quantity: 650

ISBN: 345
Title: Algorithms in C
Quantity: 650

ISBN: 345
Title: Algorithms in C
Quantity: 650

$
```

DB Pgmming: Prepared Stmts

- Problem (via another example)

Consider this SQL statement executed by order.py:

```
UPDATE orders SET quantity = quantity+1
   WHERE isbn='someisbn' AND custid='somecustid'
```

DB Pgmming: Prepared Stmts

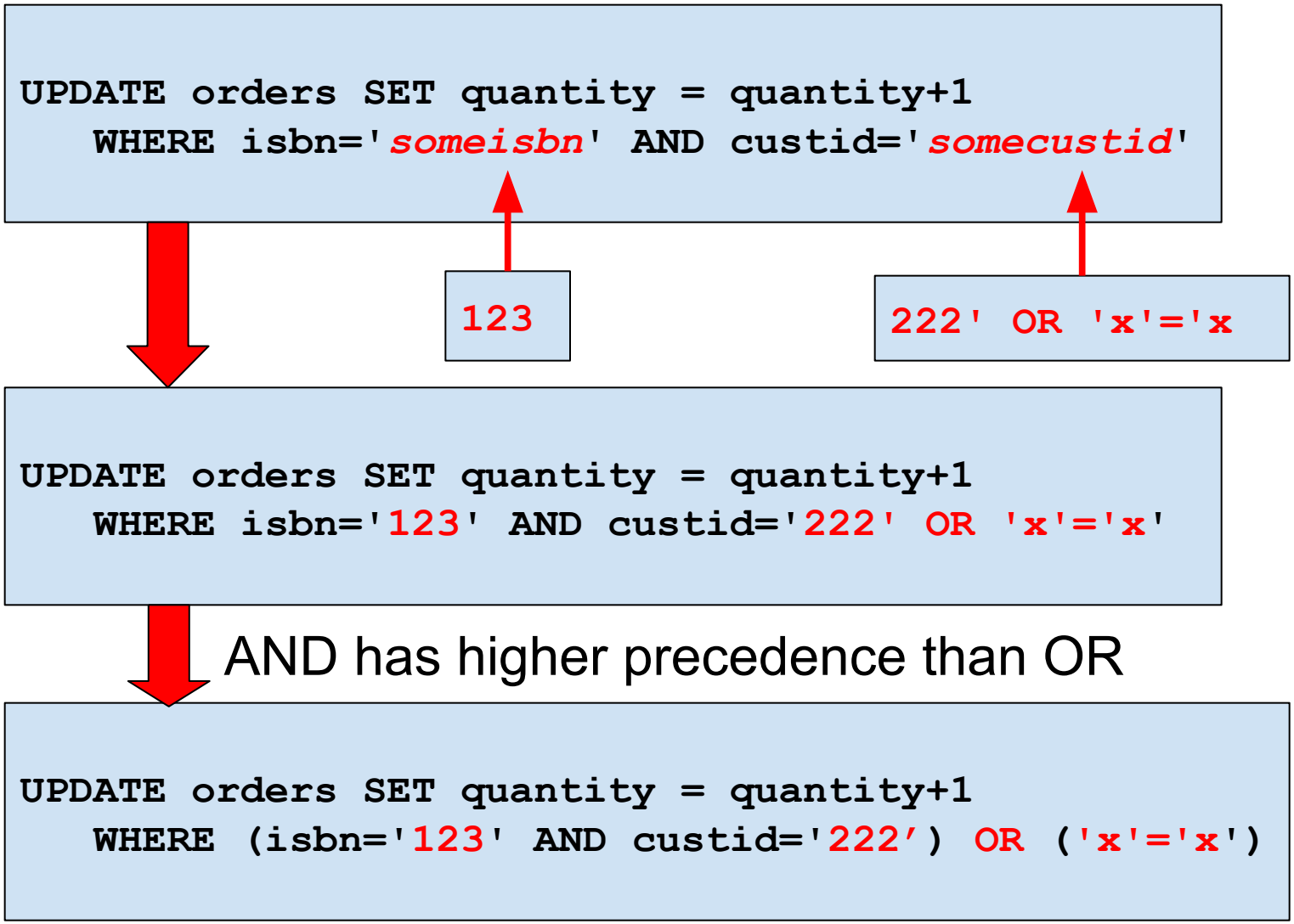
- Malicious user provides this `someisbn`

```
123
```

- … and this `somecustid` to order.py:

```
222' OR 'x'='x
```

DB Pgmming: Prepared Stmts



DB Pgmming: Prepared Stmts

```
$ python display.py
…
-----------------------------------------
orders
-----------------------------------------
('123', '222', 20)
('345', '222', 100)
('123', '111', 30)
$ python order.py 123 "222' OR 'x'='x"
$ python display.py
…
-----------------------------------------
orders
-----------------------------------------
('123', '222', 21)
('345', '222', 101)
('123', '111', 31)
```

SQL injection attack

DB Pgmming: Prepared Stmts

- For more examples of SQL injection attacks:
  - http://unixwiz.net/techtips/sql-injection.html

DB Pgmming: Prepared Stmts

- A solution...
- ***Prepared statements***

DB Pgmming: Prepared Stmts

- See **authorsearchprep.py**

```
$ python authorsearchprep.py Kernighan
ISBN: 123
Title: The Practice of Programming
Quantity: 500

ISBN: 234
Title: The C Programming Language
Quantity: 800

$
```

DB Pgmming: Prepared Stmts

- See **authorsearchprep.py** (cont.)

```
$ python authorsearchprep.py "O'Reilly"
$
```

```
$ python authorsearchprep.py "junk' OR 'x'='x"
$
```

DB Pgmming: Prepared Stmts

- See **orderprep.py**

```
$ python display.py
…
----------------------------------------
orders
----------------------------------------
('123', '222', 20)
('345', '222', 100)
('123', '111', 30)
$ python orderprep.py 123 222
$ python display.py
…
----------------------------------------
orders
----------------------------------------
('123', '222', 21)
('345', '222', 100)
('123', '111', 30)
$
```

DB Pgmming: Prepared Stmts

```
$ python display.py
…
-----------------------------------------
orders
-----------------------------------------
('123', '222', 20)
('345', '222', 100)
('123', '111', 30)
$ python orderprep.py 123 "junk' OR 'x'='x"
$ python display.py
…
-----------------------------------------
orders
-----------------------------------------
('123', '222', 20)
('345', '222', 100)
('123', '111', 30)
$
```

Summary

- We have covered:
  - Relational DB app architecture
  - Relational DB pgmming
  - Relational DB pgmming: prepared stmts